AFRL-RI-RS-TR-2007-280
Final Technical Report
January 2008

GESTURE RECOGNITION DEVELOPMENT FOR THE INTERACTIVE DATAWALL

Howard University

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2007-280 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
PETER JEDRYSIK Work Unit Manager	JAMES W. CUSACK Chief, Information Systems Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JAN 08	Final	Sep 05 – Sep 07

4. TITLE AND SUBTITLE

GESTURE RECOGNITION DEVELOPMENT FOR THE INTERACTIVE DATAWALL

5a. CONTRACT NUMBER
FA8750-05-C-0257

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)

Naren Vira

5d. PROJECT NUMBER
NASA

5e. TASK NUMBER
BA

5f. WORK UNIT NUMBER
05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Howard University
2400 6th St NW
Washington DC 20059-0002

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2007-280

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 07-0712

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Hand gestures provide a useful interface for humans to interact with not only other humans but also machines. Especially for a high degree-of-freedom manipulation tasks such as the operation of 3D objects in virtual scenes, the traditional interface composed of a keyboard and mouse is neither intuitive nor easy to operate. In collaborative environments using large screen displays for display of both 3D and 2D information, participants would benefit greatly from an interface that is unencumbered, natural, and effective for communication during discussions. The goal of this project was to investigate the feasibility of implementing an image triangulation technique to track the position of a passive device pointing towards a large screen display.

15. SUBJECT TERMS
Gesture Recognition, Interactive Display, Human-Computer Interaction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Peter A. Jedrysik
U	U	U	UL	27	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

Executive Summary

The use of hand gestures provides an attractive alternative to cumbersome interface devices for human-computer interaction. In particular, visual interpretation of hand gestures can help in achieving the ease and naturalness desired. This has motivated a very active research area concern with computer vision-based analysis and interpretation of hand gestures. To enhance multimedia capabilities in an interactive large-screen display environment such as the Air Force Research Laboratory (AFRL) DataWall [1], it is imperative to explore practical and useful gesture recognition technology. Also, due to the large screen size (12' x 3') of the DataWall, oftentimes, it is difficult to precisely identify what information presented on the screen, someone in the audience is pointing to, during professional meetings or briefings. On the same token, a presenter pointing at information on the display by hand during the presentation cannot clearly be visualized and understood in the audience. Both scenarios result into a loss of effective communication.

Implementing gesture tracking technology for the DataWall environment is a multiyear effort. The first step described in this report was to concentrate on investigating the feasibility of utilizing an image triangulation technique for accurately positioning and tracking a passive pointer pointing towards the DataWall. The pointer is marked with two distinct colors and can be tracked using two high resolution video cameras. The acquired images are then analyzed online to compute the pointer's projected coordinates on the DataWall.

1 Scope

The scope of this effort is to develop technology for building an integrated interactive display environment and intelligent interface for the Air Force Research Laboratory (AFRL) DataWall utilizing image triangulation technique to track a passive device pointing towards the DataWall.

2 Introduction

Hand gestures provide a useful interface for humans to interact with not only other humans but also machines. Especially for high degree-of-freedom manipulation tasks such as the operation of 3D objects in virtual scenes, the traditional interface composed of a keyboard and mouse is neither intuitive nor easy to operate. For such a task, we consider direct manipulation with hand gestures as an alternative method. This would allow a user to directly indicate 3D points and issue manipulation commands with his/her own hand.

The idea led to many gesture-based systems using glove-type sensing devices in the early days of virtual reality research. Such contact-type devices, however, are troublesome to put on and take off, and continuously wearing such devices for a long time fatigues users. To overcome these disadvantages vision researchers tried to develop non-contact type systems to direct human hand motion [2, 3, and 4]. These works had some instability problems particular to vision based systems. The most significant problem is occlusion. Vision systems conventionally require match of detected feature points between images to reconstruct 3D information. However, for moving non-rigid objects like a human hand, detection and matching of feature points is difficult to accomplish correctly.

Providing a computer with the ability to interpret a human hand is a step toward more natural human-machine interactions. Existing input systems augmented with this, as well as such other human-like modalities such as speech recognition and facial expression understanding, will add a powerful new dimension to the range of future computer applications and the accessibility of existing ones. A wide spectrum of research is underway on the problem of gesture interpretation. The primary reason for the advancement is continuously falling expense of hardware and image grabbing and processing. Even color processing in now available and it is fast enough for pattern recognition.

Currently there is no universal definition of what a gesture recognition system should do or even what is a gesture. Our definition of gesture form perspective of the computer is simply a temporal sequence of images of a hand. An element from a finite set of static hand poses is the expected content with an image

frame. A gesture is, therefore, a sequence of static hand poses. Poses are assumed to contain the identity of the hand shape and (possibly) the orientation, translation and distance from camera information. The spatio-temporal nature of the gesture data make the gesture state immeasurable at a given instance in time, but for each time step we can determine the static hand pose. A general gesture recognition system is depicted in Figure 1. Visual images of gestures are acquired by one or more cameras. They are processed in the analysis stage where the gesture model parameters are estimated. Using the estimated parameters and some higher level knowledge, the observed gestures are inferred in the recognition stage. The grammar provides a set of rules on which the gestures are interpreted.

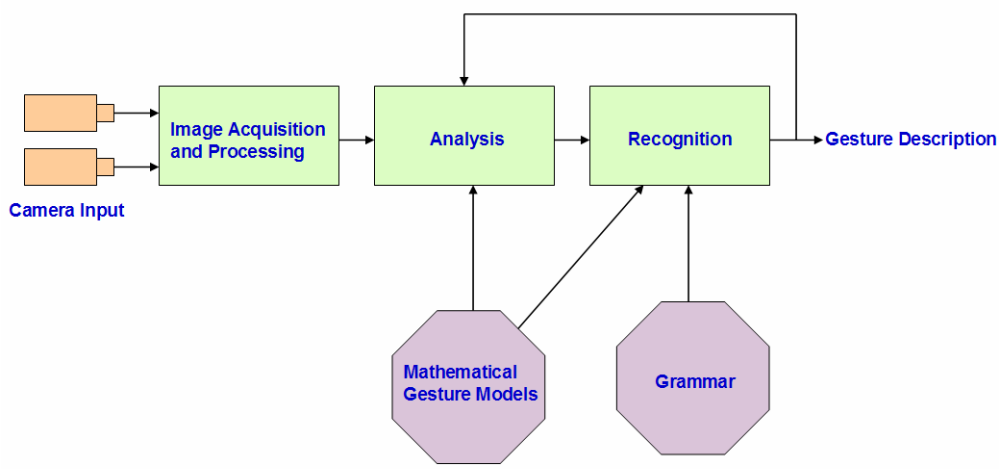

Figure 1 Gesture Interpretation System

The project of developing and implementing gesture tracking technology for the interactive DataWall is an ambitious project and will take several years of effort. It encompasses several major steps which can be grouped into two major categories:

A. Recognition and tracking of a color pointer
B. Recognition and tracking of a hand gesture

This work was devoted towards category A in which a passive color pointer marked with two distinct colors will be tracked using two high resolution cameras. The work can alternatively be viewed as the development of virtual pointer technology as opposed to commonly used laser pointer. The methodology is described in the following section.

3 Recognition and Tracking a Passive Pointer

3.1 Technical discussion

The procedure for recognition and tracking of a passive pointer marked with two distinct colors is described in this section based on the theory of image processing and analysis. Consider a system with two cameras of focal length f and baseline distance b as shown in Figure 2. The optical axes of the two cameras are converging with an angle θ and that all geometrical parameters (b, f, and θ) are known or estimated using a camera calibration technique [5-8]. A feature in the scene depicted at the point P is viewed by the two cameras at different positions in the image planes (I_1 and I_2). The origins of the each camera coordinate system is located at the camera's center which is distance f away form the corresponding image planes I_1 and I_2, respectively. It is assumed, without loss of generality, that the world coordinate system (Cartesian coordinates X, Y, and Z) coincides with the coordinate system of camera 1 (left camera), while the coordinate system of camera 2 (right camera) is obtained from the former through rotation and translations. The plane passing trough the camera centers and the feature point in the scene is called the epipolar plane. The intersection of the epipolar plane with the image plane defines the epipolar line as shown in Figure 3. For the model shown in the figure, every feature in one image will lie on the same row in the second image. In practice, there may be a vertical disparity due to misregistration of the epipolar lines. Many formulations of binocular stereo algorithms assume zero vertical disparity.

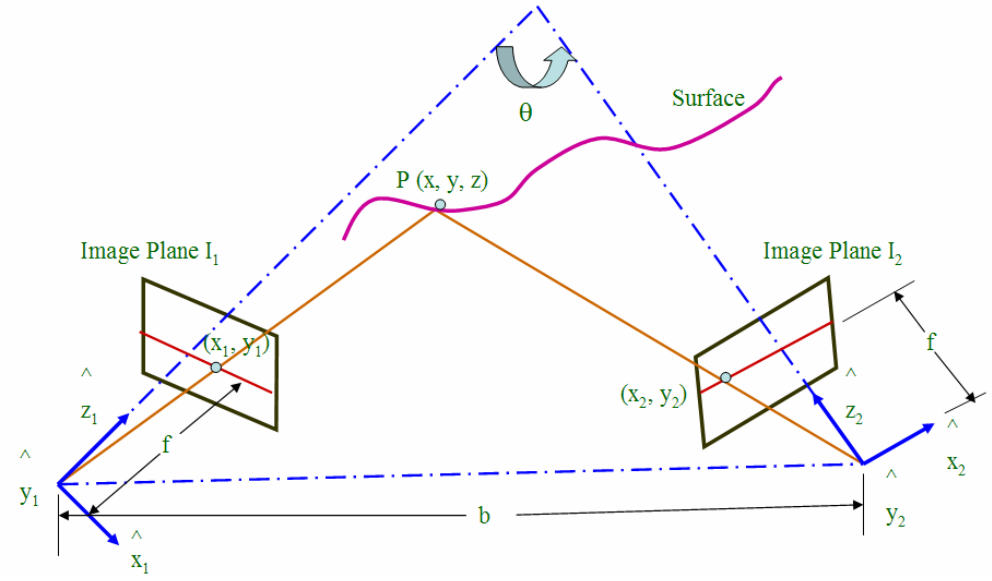

Figure 2 Non-Parallel Axes Camera Model

The point P with the world coordinates (X, Y, and Z) is projected on image plane I_1 as point (x_1, y_1) and image plane I_2 as point (x_2, y_2) as illustrated in Figure 2. Then, assuming a perspective projection scheme, a simple relation between the camera coordinates (x_1, y_1) and world coordinates (X, Y, and Z) can be obtained as

$x_1 = f * X / Z$ and $y_1 = f * Y / Z$ (1)

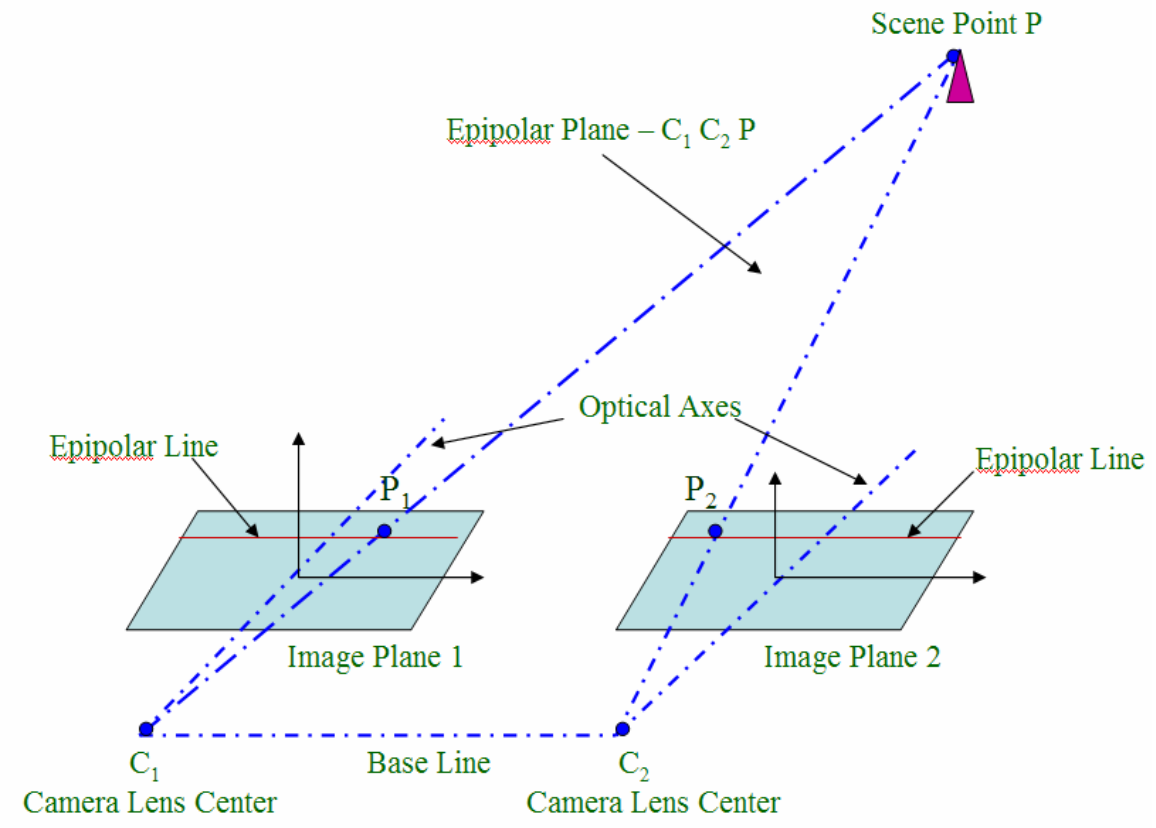

Figure 3 The Epipolar Plane

Similarly, we can write

$x_2 = f * x_2\hat{} / z_2\hat{}$ and $y_2 = f * y_2\hat{} / z_2\hat{}$ (2)

Where, coordinate system of camera 2 ($x_2\hat{}$, $y_2\hat{}$ and $z_2\hat{}$) is related with respect to the world coordinate system by simply translation and rotation as

$$x_2^{\wedge} = c\,X + s\,Z - b\,c'$$
$$y_2^{\wedge} = Y \qquad (3)$$
$$z_2^{\wedge} = -s\,X + c\,Z + b\,s'$$

Here, symbols $c = \cos(\theta)$ and $s = \sin(\theta)$, $c' = \cos(\theta/2)$ and $s' = \sin(\theta/2)$ are used. Substituting Eq. (3) into Eq. (2), we can write

$$x_2 = f\,[(c\,X + s\,Z - b\,c') / (-s\,X + c\,Z + b\,s')] \qquad (4)$$
$$y_2 = f\,[Y/(-s\,X + c\,Z + b\,s')]$$

Combining Eq. (1) and Eq. (4), lead to

$$x_2 = f\,[(f\,s + x_1\,c)Z - f\,b\,c'] / [(f\,c - x_1\,s)Z + f\,b\,s'] \qquad (5)$$
$$y_2 = (f\,Z\,y_1) / [(f\,c - x_1\,s)Z + f\,b\,s']$$

It can be observed for Eq. (5) that the depth Z of P can be estimated if its projections (x_1, y_1) and (x_2, y_2) on image planes I_1 and I_2, respectively, are known. That is for a given point (x_1, y_1) on I_1, its corresponding point (x_2, y_2) on I_2 should be found. Hence, defining a disparity vector $\mathbf{d} = [dx, dy]^T$ at location (x_2, y_2) of camera 2 with respect to camera 1

$$dx = x_1 - x_2 \qquad (6)$$
$$= \frac{f\,b\,(f\,c' + x_1\,s') + [\,x_1(f\,c - x_1 s) - f\,(f\,s + x_1 c)\,]Z}{(f\,c - x_1\,s)Z + f\,b\,s'}$$

$$dy = y_1 - y_2 \qquad (7)$$
$$= \frac{f\,b\,s'\,y_1 + [\,(f\,c - x_1\,s) - f\,]\,y_1\,Z}{(f\,c - x_1\,s)Z + f\,b\,s'}$$

If the disparity vector **d** is known, Eqs. (6-7) reduce to an over determined linear system of two equations with a single unknown, Z (the depth) and a least-squares solution can be obtained [9]. When cameras axes are parallel (i.e., $\theta = 0$) the above equations (Eqs. (6-7)) can be simplified to (see Ref. [10] and Fig. 5)

$$d_x = f\,b / Z \text{ and } d_y = 0 \qquad (8)$$

Thus, the depth at various scene points may be recovered by knowing disparities of corresponding image points.

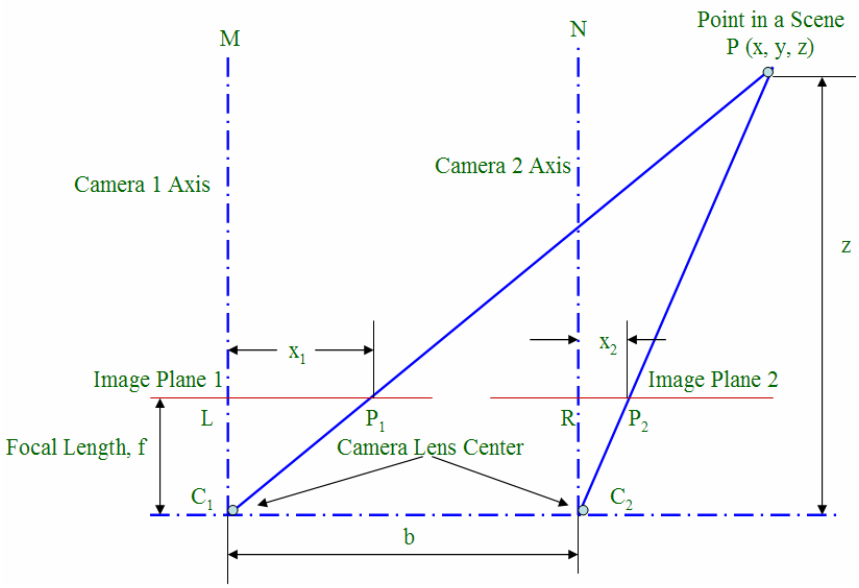

Figure 4 The Parallel Axes Camera Model

3.2 Color Representation and Detecting Two-Color Ends of a Pointer

Each pixel in RGB color space can be expressed in a vector form as

$$\mathbf{P}(I, J) = R(I, J)\,\mathbf{i} + G(I, J)\,\mathbf{j} + B(I, J)\,\mathbf{k} \quad (9)$$

The image pixel coordinates are (I, J) and **i**, **j**, and **k** are unit vectors along R, G, and B color space, respectively. Since we are only interested in matching the pointer's red and blue color ends of each image respectively, Equation (9) can be simplified as

$$\mathbf{P}(I, J) = R(I, J) \quad (10)$$

when the red color end is considered and

$$\mathbf{P}(I, J) = B(I, J) \quad (11)$$

for the blue color end. Note that **P** (I, J) is mathematically scalar quantity. We can now scan each image to find all pixels and corresponding locations for particular color end. We compute the centroid of each color end. That is for the red color end as shown in Figure 5, image 1 (left), we have

$\mathbf{P_1}$ (I, J) = R_1 (Imid, Jmid) (12)

Where,

Imid = Imin + (Imax – Imin)/2

Jmid = Jmin + (Jmax – Jmin)/2

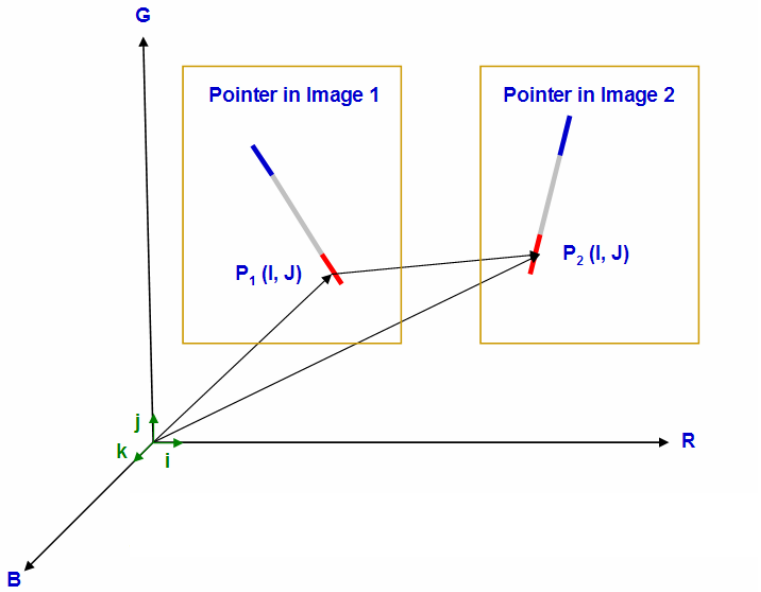

Figure 5 Color Vector Representation in RGB Space of Matching Pixels in Two Different Images

The terms mid, min and max correspond to the mid point, minimum location and maximum location of the color within that particular color end. Note that the image has to be searched to find the min and max locations. The term centroid and mid point of the color end are interchangeable because of the two-dimensional coordinate system representation. Similarly, we can compute the centroid of the red color end in image 2 (right) as

$\mathbf{P_2}$ (x, y) = R_2 (Imid, Jmid) (13)

We assume that the centroid points $\mathbf{P_1}$ (I, J) and $\mathbf{P_2}$ (I, J) represent the matching points. This assumption is valid because the pointer dimensions are very small in comparison with the dimension of the DataWall room. Note the image size. Thus, the implication is that the process of disparity analysis is not required and the task of finding matching pixel is considerably simplified. The same analysis can be applied for finding the matching points corresponding to the blue color end of

the pointer. It should be emphasized that we deliberately chose two distinct color-ends to simplify and speed up the process of image scanning. One can choose other pixel matching methods depending upon their application. Knowing the x- and y- coordinates of each centroid point of the pointer in a single image; we can mathematically pass a line through these two points to describe a pointer in a 2D space. Now the process of triangulation in needed to compute the three-dimensional coordinates of the pointer from these two images (i.e., four centroid points).

3.3 Three-dimensional Triangulation Technique

We apply ray casting analysis to triangulate three-dimensional coordinates of each image pixel point in a space as it viewed by two cameras with respect to a chosen reference frame. Without loss of generality, the reference fame could be at one of the cameras' center. We have chosen camera 2 center location as the frame of reference. Each ray is cast from the viewpoint (here, center of the camera) through each pixel of the projection plane (here, image planes 1 and 2) into the volume dataset. The two rays wherever they intersect in a 3D space determines the coordinates of a point viewed in both camera as shown in Figure 6. By connecting all intersecting points in the volume dataset, we can generate a 3D point cloud floating in space. We utilize only four points (two in each image) to find the 3D position of the pointer.

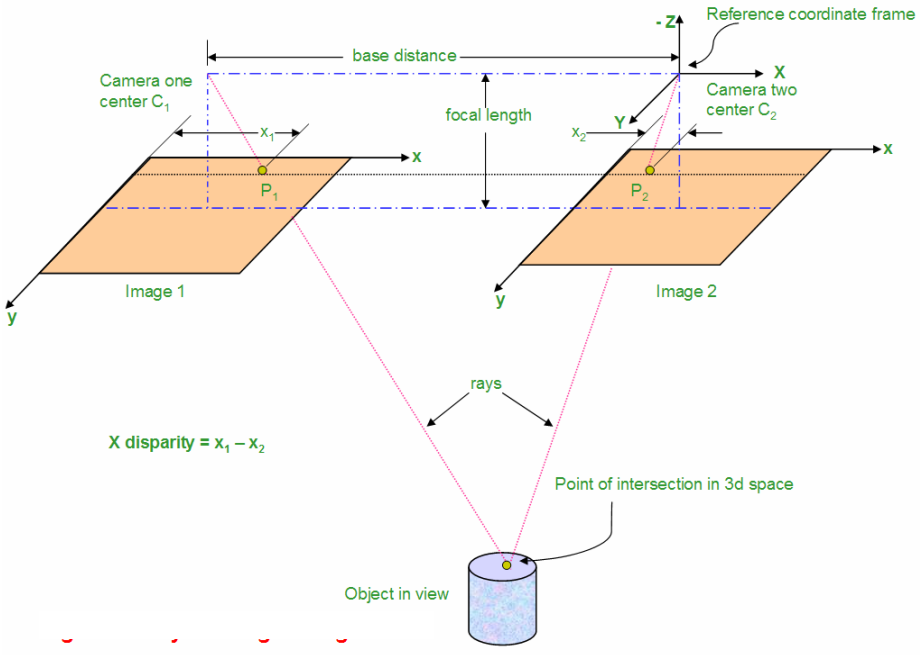

Figure 6 Ray Casting Configuration

3.4 Two Intersecting Line Problem

The common point coordinate computation of rays reduces to a problem of two line intersection each defined by two points. One point on the line is defined by the camera center and the second point by a pixel in the image plane (i.e. P_1 or P_2 in Figure 7). For the point P_1 of image 1, the coordinates of point P_2 in image 2 are already chosen based on the explanation presented earlier.

Considering a general reference frame (x, y, z) as shown in Figure 7, point sets (C_1, P_1) and (C_2, P_2) are situated on line1 and 2, respectively. Since the points P_1 (I, J) and P_2 (I, J) are in pixel coordinates, they need to be converted into linear measurements by the transformation:

$$\text{x distance per pixel} = \frac{\text{f *tan (half view angle of camera)}}{\text{(Image width in pixel) / 2}} \qquad (14)$$

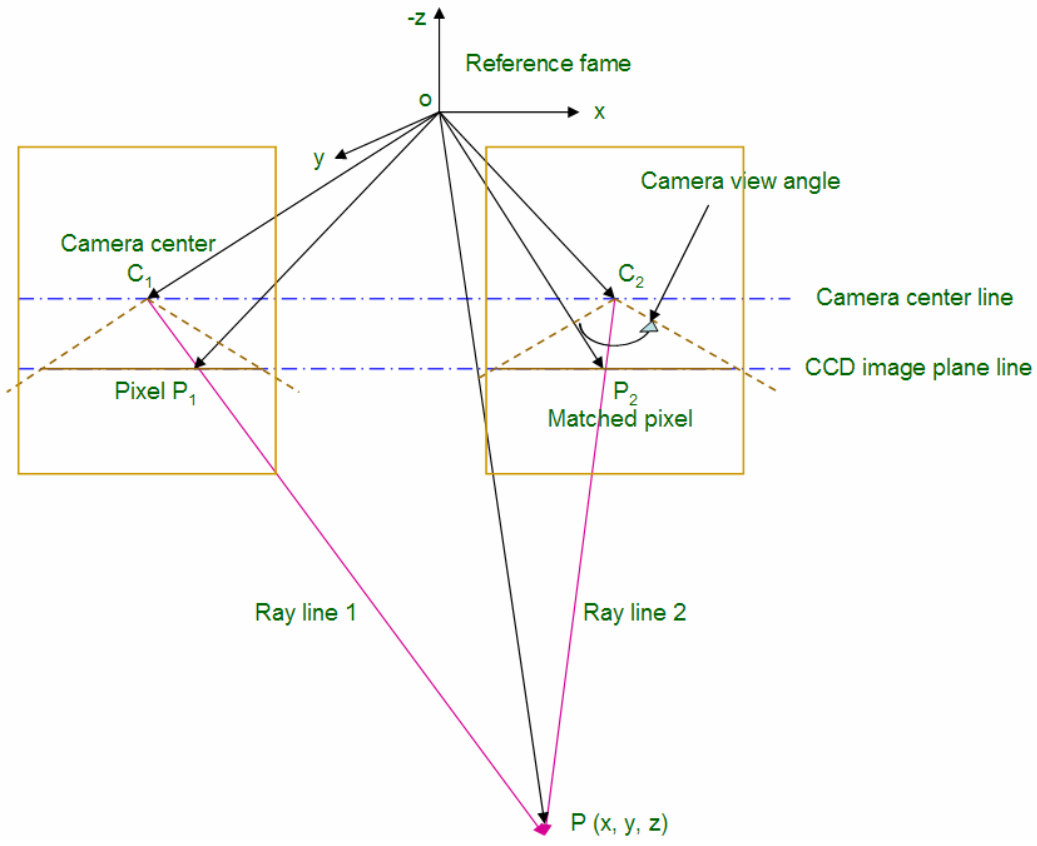

Figure 7 Coordinate Computation for Two Lines of Intersection

Similarly, y distance per pixel can be correlated. Note that f denotes camera focal length. Because we are interested in computing coordinates of point P, let us define each point on the lines as

$$\mathbf{P} = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} \quad (15)$$
$$\mathbf{P_1} = P_{x1}\,\mathbf{i} + P_{y1}\,\mathbf{j} + P_{z1}\,\mathbf{k}$$
$$\mathbf{P_2} = P_{x2}\,\mathbf{i} + P_{y2}\,\mathbf{j} + P_{z2}\,\mathbf{k}$$
$$\mathbf{C_1} = C_{x1}\,\mathbf{i} + C_{y1}\,\mathbf{j} + C_{z1}\,\mathbf{k}$$
$$\mathbf{C_2} = C_{x2}\,\mathbf{i} + C_{y2}\,\mathbf{j} + C_{z2}\,\mathbf{k}$$

Where **i**, **j**, and **k** are unit vectors along x, y and z axes, respectively. With the condition for the four points to be coplanar (the lines are not skewed), we can write

$$(\mathbf{C_2} - \mathbf{C_1}) \bullet [(\mathbf{P_1} - \mathbf{C_1}) \times (\mathbf{P_2} - \mathbf{C_2})] = 0 \quad (16)$$

Where symbols • and x represent vector dot and cross product respectively. If s and t are scalar quantities then the common point can be represented parametrically as

$$\mathbf{P} = \mathbf{C_1} + s\,(\mathbf{P_1} - \mathbf{C_1}) = \mathbf{C_1} + s\,\mathbf{A} \quad (17)$$

or

$$\mathbf{P} = \mathbf{C_2} + t\,(\mathbf{P_2} - \mathbf{C_2}) = \mathbf{C_2} + t\,\mathbf{B}$$

Where s is given by

$$s = \frac{[(\mathbf{C_2} - \mathbf{C_1}) \times \mathbf{B})] \bullet (\mathbf{A} \times \mathbf{B})}{|\,\mathbf{A} \times \mathbf{B}\,|^2}$$

3.5 Accounting for a Camera Rotations

Six degrees-of-freedom are required to describe a point in the three-dimensional space uniquely. One can choose three linear and three rotational coordinates. The three rotational motions of the camera can be accounted for while computing uniquely the pointer's position in the 3D space. Defining each camera's axis rotation as pitch, yaw and roll along x, y and z axes, respectively, as shown in Figure 8, we can write rotational transformations as

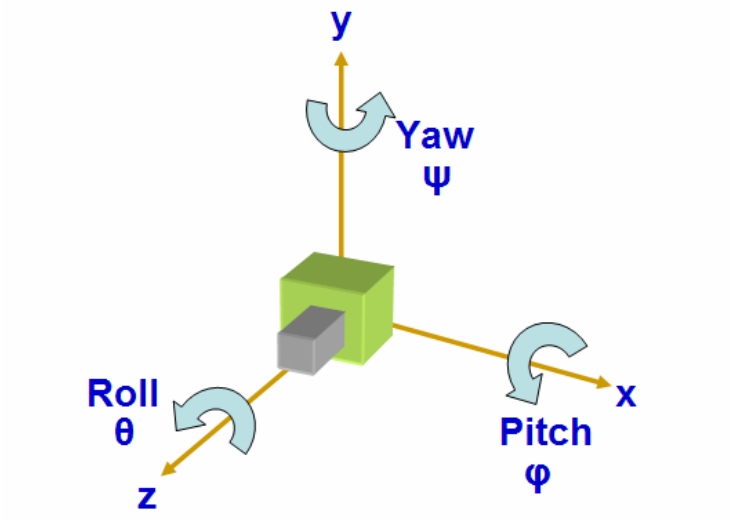

Figure 8 Camera's Pitch, Yaw, and Roll Axes

$$R(x, \text{pitch}) = R(x, \varphi) = \begin{Bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{Bmatrix} \quad (18)$$

$$R(y, \text{yaw}) = R(y, \psi) = \begin{Bmatrix} C\psi & 0 & S\psi \\ 0 & 1 & 0 \\ -S\psi & 0 & C\psi \end{Bmatrix} \quad (19)$$

$$R(z, \text{roll}) = R(z, \theta) = \begin{Bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{Bmatrix} \quad (20)$$

Where, the notations S(angle) = sin (angle) and C(angle) = cos (angle) are used. The combined transformation pitch-yaw-roll can be written as PYR

$$\begin{aligned} PYR &= R(x, \text{pitch})\, R(y, \text{yaw})\, R(z, \text{roll}) \\ &= R(x, \varphi)\, R(y, \psi)\, R(z, \theta) \end{aligned} \quad (21)$$

$$= \begin{Bmatrix} C\theta\, C\psi & -S\theta\, C\psi & S\psi \\ C\theta\, S\varphi S\psi + C\varphi\, S\theta & C\theta\, C\varphi - S\theta\, S\varphi\, S\psi & -C\psi\, S\varphi \\ S\theta\, S\varphi - C\theta\, C\varphi\, S\psi & S\theta C\varphi\, S\psi + C\theta S\varphi & C\varphi\, C\psi \end{Bmatrix}$$

The world coordinates (x, y, z) are, thus, related to camera's view coordinates (x', y', z') as

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \{PYR\} \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \qquad \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \{PYR^{-1}\} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \quad (22)$$

Note that inverse transformation is considered to account for the camera rotations.

3.6 Point of Projection on the DataWall

Knowing the three-dimensional coordinates of each end of the pointing device center (red and blue), we can identify and represent the pointer in a 3D space by a line passing through these two points. The pointing device passing through the points P_r and P_b as depicted in Figure 9. The projection of this line on a plane described by DataWall is of our interest. The problem is now reduced to finding coordinates of intersecting point between line and a plane as shown by point P_i in Figure 9.

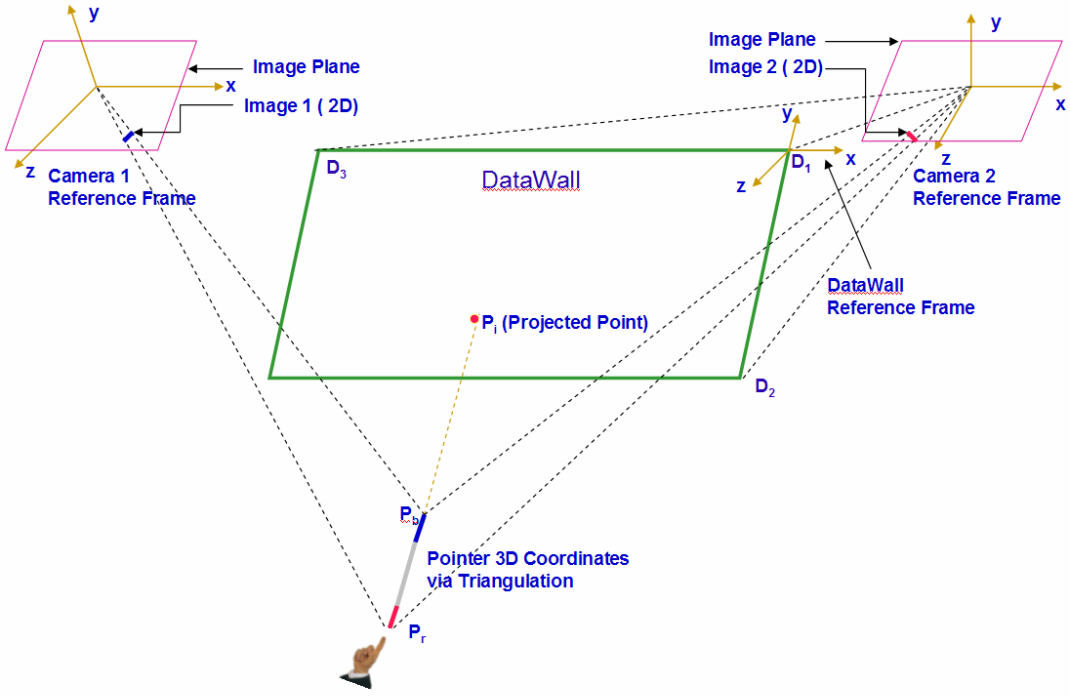

Figure 9 Three-Dimensional Pointer Projection on Datawall

3.7 Equation of a Plane Describing the DataWall

The standard equation of a plane in a 3D space is:

$$Ax + By + Cz + D = 0 \qquad (23)$$

Where, the normal to the plane is the vector (A,B,C). Given three points in space D_1(x1,y1,z1), D_2(x2,y2,z2), D_3(x3,y3,z3) the equation of the plane through these points is given by the following determinants.

$$A = \begin{vmatrix} 1 & y1 & z1 \\ 1 & y2 & z2 \\ 1 & y3 & z3 \end{vmatrix} \quad B = \begin{vmatrix} x1 & 1 & z1 \\ x2 & 1 & z2 \\ x3 & 1 & z3 \end{vmatrix} \quad C = \begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix} \quad D = - \begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} \qquad (24)$$

Here, three points D_1, D_2 and D_3 describes the DataWall referenced in the camera 2 coordinate system. Expanding the above gives

$$A = y1 (z2 - z3) + y2 (z3 - z1) + y3 (z1 - z2) \qquad (25)$$
$$B = z1 (x2 - x3) + z2 (x3 - x1) + z3 (x1 - x2)$$
$$C = x1 (y2 - y3) + x2 (y3 - y1) + x3 (y1 - y2)$$
$$D = - [x1 (y2 z3 - y3 z2) + x2 (y3 z1 - y1 z3) + x3 (y1 z2 - y2 z1)]$$

Note that if the points are colinear then the normal (A,B,C) as calculated above will be (0,0,0). The sign of s = Ax + By + Cz + D determines which side the point (x,y,z) lies with respect to the plane. If s > 0 then the point lies on the same side as the normal (A,B,C). If s < 0 then it lies on the opposite side, if s = 0 then the point (x,y,z) lies on the plane.

3.8 Intersection of Line and Plane

The parametric representation of the equation of the line passing through points **Pr** (rx, ry, rz) and **Pb** (bx, by, bz) is made as

$$P = Pr + u (Pb - Pr) \qquad (26)$$

Where, Pr and Pb are the center of red and blue color ends of the pointing device. The point of intersection of the line and plane can be found by solving the system of equations defined above (i.e., Eqs (23) and (26)). That is

$$A (rx + u (bx - rx)) + B (ry + u (by - ry)) + C (rz + u (bz - rz)) + D = 0 \quad (27)$$

Solving for u

$$u = \frac{A*rx + B*ry + C* rz + D}{A (rx – bx) + B(ry – by) + C(rz – bz)} \quad (28)$$

Now plug it back into the equation of the line to get the point of intersection, P_i defined in Figure 9. It is reminded that when the denominator is 0 in u then the normal to the plane is perpendicular to the line. Thus the line is either parallel to the plane and there are no solutions or the line is on the plane in which case are infinite solutions.

3.9 Recognition of a Hand Gesture

Hand gestures can be classified into two classes: (1) static hand gestures which relies only on the information about the angles of the figures (hand posture) and (2) dynamic hand gestures which relies not only on the fingers' flex angle but also the hand trajectories and orientations. In general, a hand gesture is expressed as a time series of hand position, orientation, and shape. Hand shape is most difficult to recognize, though, how it is recognized depends on how it is utilized. Since our goal is to develop a non-contact hand gesture recognizer which can be utilized in a virtual environment, it is sufficient to discriminate from among only a few typical hand shapes, such as the number of extended fingers, as graphical commands.

For gesture interpretation system, there are four main components: gesture modeling, gesture analysis, gesture recognition and gesture based application systems. The fist phase of a recognition task (whether considered explicitly or implicitly) is choosing a model of the gesture. The mathematical model may consider both the spatial and temporal characteristic of the hand and hand gesture. Once the model is decided upon, an analysis stage is used to compute the model parameters form input image features. These parameters constitute some description of the hand pose or trajectory and depend on the modeling approach used. Among the important problems involved in the analysis are those of hand localization, hand tracking, and selection of suitable image features. The computation of model parameters is followed by gesture recognition. Here, the parameters are classified and interpreted in the light of the accepted model and perhaps the rules imposed by some grammar. Evaluation of a particular gesture recognition approach encompasses accuracy, robustness, and speed, as well as the variability in the number of different classes of hand/arm movements it covers.

4 Computer Simulation

The feasibility of utilizing an image triangulation technique for accurately positioning and tracking a virtual pointer pointing towards DataWall was investigated. The modeling and simulation task was carried out in which synthetic images of the pointer (generated using Autodesk® 3ds Max®) were input to a Microsoft® Visual C^{++} program. Based on the theory described in the previous section a Visual C^{++} program was written which requires two cameras' images as an input and determines the 3D coordinates of the pointer as well as the pointer's pointing projection on the DataWall. The analysis is done on high resolution static images utilizing different room configurations. The projected locations of a virtual pointer on the DataWall were compared with the known locations retrieved from the 3ds Max® models. The results were promising and the pointing accuracy of the pointer on the DataWall was in the neighborhood of 0.06 feet. This accuracy is regarded to be well within acceptable range.

Figure 10 describes various reference frames defined for testing the present methodology. The output results of the C++ algorithms are divided into three groups. One, the pointer's pointing position accuracy on the DataWall without rotating any cameras; two, when camera rotations are included in the analysis; and three when pointer's length variations are considered. Table 1 presents five different scenarios for the group one. The highlighted pink area describes changes in the configuration with respect to the case # 1. The output of the algorithm (the pointer's projection on the DataWall) using triangulation method is compared with the corresponding retrieved values from 3ds Max® program.

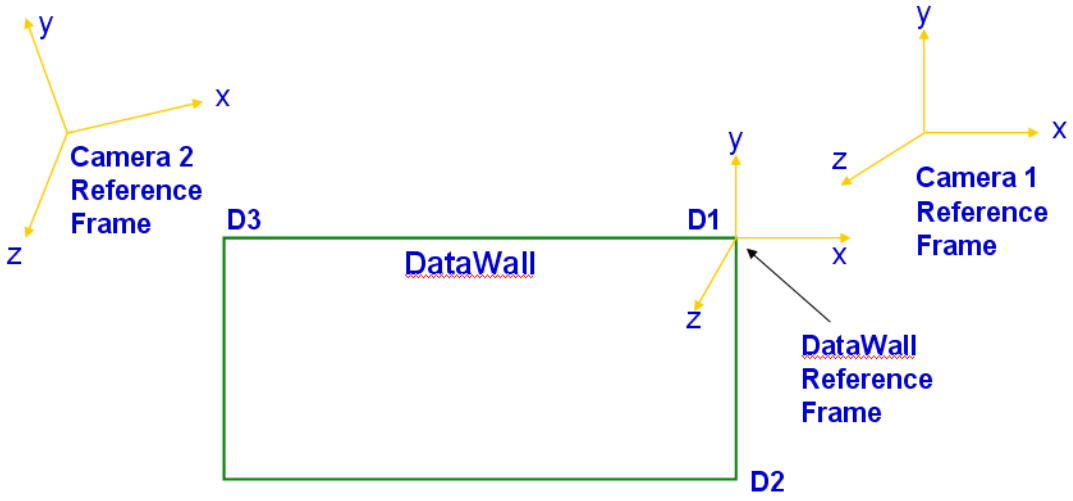

Figure 10 Definition of Reference Frames for Testing

The worse case scenario is off by 0.041 feet in y coordinate. The absolute average for all five cases is 0.006 feet and 0.034 feet in the x- and y-coordinates, respectively. These accuracies are considered reasonable for the specified goals. The simulation results are tabulated in Table 1.

Table 1 Positioning Accuracy Comparison

Test Case	Camera 1 Position			Camera 2 Position			DataWall Position									Actual DataWall Projection - Studio 3DMax		Computed DataWall Projection		Difference in Position	
							Point D1			Point D2			Point D3								
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	x	y	x	y
	pitch	yaw	roll	pitch	yaw	roll															
1	-12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-5.999	1.663	-0.001	0.037
	0.0	0.0	0.0	0.0	0.0	0.0															
2	-12.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	1.7	-5.983	1.738	-0.017	-0.038
cam	0.0	0.0	0.0	0.0	0.0	0.0															
shift																					
5	-12.0	0.0	0.0	0.0	0.0	0.0	-0.37	2.745	0.583	-0.45	3.939	-2.71	-11.5	-1.34	-0.61	-6.0	1.7	-6	1.659	0.000	0.041
DW	0.0	0.0	0.0	0.0	0.0	0.0															
rotate																					
8	-12.0	0.0	0.0	0.0	0.0	0.0															
pointer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-6.0	-1.7	-5.999	-1.663	-0.001	-0.037
move																					
9	-12.0	0.0	0.0	0.0	0.0	0.0															
pointer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-3.5	0.0	-12.0	0.0	0.0	-5.0	-3.0	-5.012	-2.984	0.012	-0.016
move																					
	All dimensions are in feet. Highlighted area describes the changes with respect to case # 1																Absolute Average			0.006	0.034

5 Camera Calibration

The camera image quality should be high enough for the proposed project methodology to work. Commercially available video cameras capable of capturing images of 1920 x 1080 pixels at 60 frames per second were used. The cameras were a very new product at the time with a limited user interface for configuration. As a result there was some difficulty getting an acceptable image output from the cameras. The supplier was contacted and per their suggestion, the camera's processing system was configured in a HyperTerminal mode. With many arbitrary trials, we were successful in getting improved images (see Figure 11 below).

Figure 11 Image Acquired after Camera Calibration (1920 x 1080 pixels)

6 Image Acquisition and API Development

6.1 Hardware Setup

The system was configured to acquire two camera images simultaneously by installing two frame grabbers, X64-CL_iPro in a Dell 470 workstation. The cable connection to the frame grabber is shown in Figure 12 below.

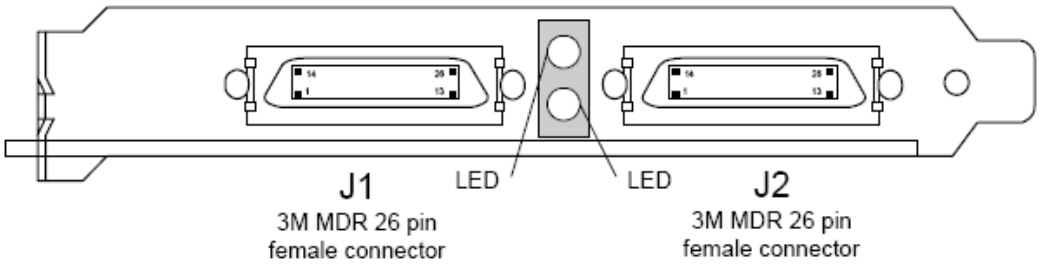

Figure 12 X64-CL iPro Frame Grabber Cable Connection

DALSA Coreco's Sapera LT 5.3 software was installed on the computer system. Furthermore, all other necessary application programs were installed. The API described below was developed based on Microsoft's Visual C^{++} NET 2003.

6.2 API Deployment

The TwoCam – Stage 1 API views two cameras simultaneously when they are attached to two different frame grabbers. The program grabs images from a camera into a buffer in the host computer's memory using Sapera LT ++ Acquisition and Buffer objects and then Transfer object to link them. Also, a View object is used to display the buffer.

For each camera class the following objects were created:

Acquisition object
Buffer object
Transfer object
View object

Note that separate class is needed for each camera. The program runs in a continuous mode via XferCamera = Grab ( ) object statement. If we use XferCamera = Snap ( ) object statement, the program snaps the view scene and terminates. This program mode is useful for static analysis.

The program generates two outputs. One, specific parameters that were utilized in run mode are displayed in a command window as shown below (Figure 13):

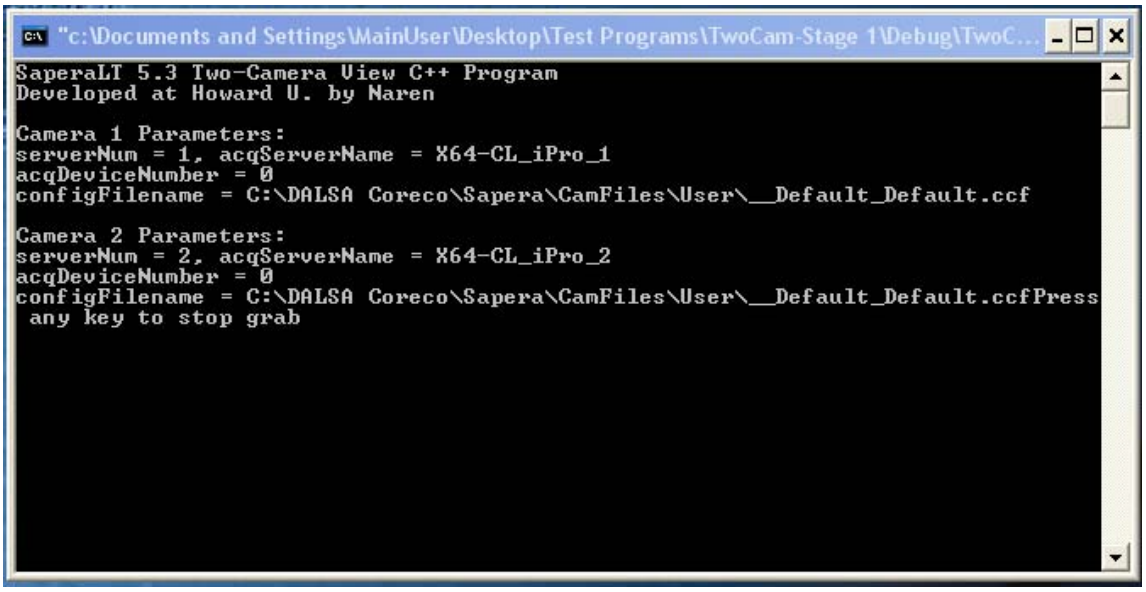

Figure 13 Command Output Window

The second output is dumped in a text file for further analysis and use. The program also displays two viewing windows. The camera viewing window for each camera is shown below (Figure 14). Here, a “test pointer” is being viewed simultaneously with both cameras.

Figure 14 Acquisition of Two Simultaneous Images from Two Different Cameras

7 Conclusion

An initial attempt was made with some success to develop an API for triangulating two images in order to track the pointing position of a passive pointer pointing toward the DataWall screen. Images are acquired and displayed simultaneously. The next step will require each image pixel to be split up into RGB color for further analysis.

References:

[1] P. Jedrysik and R. Alvarez, Advanced Displays and Intelligent Interfaces (ADII), AFRL-IF-RS-TR-2006-230, July 2006.

[2] B.Moghaddam and A. Pentland, “Maximum Likelihood of Detection of Faces and Hands,”Proceeding of International Workshop on Automatic Face and Gesture Recognition, 1995, pp. 122-128.

[3] P.Cipolla and N. Hollinghurst, “Uncalibrated Stereo Vision with Pointing for a Man-Machine Interface,” Proceedings of IAPR Workshop on Machine Vision Applications, 1994, pp. 163-166.

[4] L. Gupta and S. Ma, “Gesture-Based Interaction and Communication: Automated Classification of Hand Gesture Contours,” IEEE Transactions on System, Man an Cybernetics – Part C: Applications and Reviews, Vol. 31, No.1, Feb 2001, pp.114 – 120.

[5] R. Tsai, “An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL., 1986, pp. 364 – 374.

[6] O.Faugeras and G. Toscani, “The Calibration Problem for Stereo, International Proceedings of CVPR, 1986, pp. 15 –20.

[7] B. Caprile and V. Torre, “Using Vanishing Points for Camera Calibration,” International Journal of Computer Vision, Vol. 3, 1990, pp. 127 – 140.

[8] S. Maybank and O. D. Faugeras, “A Theory of Self-Calibration of a Moving Camera,” International Journal of Computer Vision, Vol. 8, 1992, 123 – 151.

[9] D. Tzovaras, N. Grammalidis and M. G. Stromtzis, “Object-based Coding of Stereo Image Sequences using Joint 3D Motion / Disparity Compensation,” IEEE Transaction on Circuits System Video Tech., Vol. 7, April 1997, pp. 312 – 327.

[10] R.Jain, R. Kasturi and B. Schunck, Machine Vision, Book, McGraw Hill, Inc., 1995.

[11] N. Vira, “3D Stereoscopic Image Reconstruction,” Research Report to U. S. Air Force,August 2003, Contract No. 28459.

[12] N. Vira, “Use of JView in 3D Digital Color Image Reconstruction and Visualization,”ASME Design Engineering Technical Conferences and Computers

and Information Engineering Conference, Salt Lake City, Utah, September 28 – October 2 2004, DETC2004 – 57668, pp. 1 –10.